**Task 21 to Task 30 – home tasks — plz refer Doc 17 Arrays in java..**

**Final command is used to create constant**

**Task 031**

**class Calculation{**

**int z;**

**public void addition(int x, int y) {**

**z = x + y;**

**System.out.println("The sum of the given numbers:"+z);**

**}**

**public void Subtraction(int x, int y) {**

**z = x - y;**

**System.out.println("The difference between the given numbers:"+z);**

**}**

**}**

**public class My\_Calculation2 extends Calculation {**

**public void multiplication(int x, int y) {**

**z = x \* y;**

**System.out.println("The product of the given numbers:"+z);**

**}**

**public static void main(String[] args) {**

**int a = 20, b = 10;**

**My\_Calculation2 obj = new My\_Calculation2();**

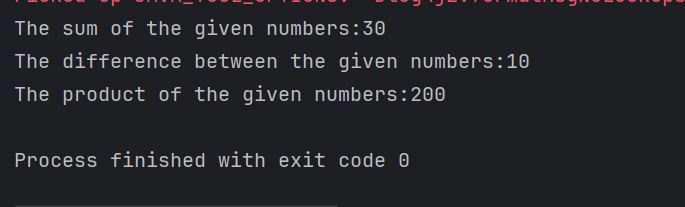
**obj.addition(a, b);**

**obj.Subtraction(a, b);**

**obj.multiplication(a, b);**

**}**

**}**

****

**Task 032**

**class Calculations {**

**int z;**

**public void addition(int x, int y) {**

**z = x + y;**

**System.out.println("Sum: " + z);**

**}**

**public void subtraction(int x, int y) {**

**z = x - y;**

**System.out.println("Difference: " + z);**

**}**

**}**

**// Clock as an interface**

**interface Clock {**

**void showTime();**

**}**

**public class task32 extends Calculations implements Clock {**

**public void multiplication(int x, int y) {**

**z = x \* y;**

**System.out.println("Product: " + z);**

**}**

**// Implementing Clock method**

**public void showTime() {**

**System.out.println("Current time: 10:30 AM");**

**}**

**public static void main(String[] args) {**

**task32 obj = new task32();**

**obj.addition(10, 5);**

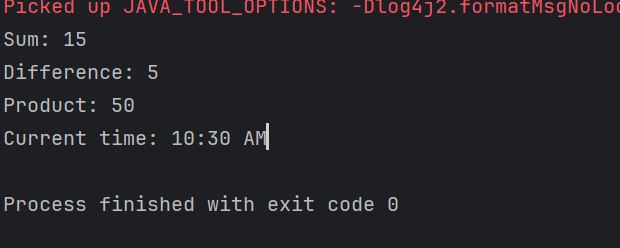
**obj.subtraction(10, 5); // ✅ corrected here**

**obj.multiplication(10, 5);**

**obj.showTime();**

**}**

**}**

****

**Task 33**

**class Customer{**

**int cost = 40;**

**String items = "Tomatoes";**

**Customer(){**

**System.out.println("Constructor called");**

**}**

**void purchage\_list(){**

**System.out.println("cost of tomatoes in Customer class is "+ cost);**

**}**

**}**

**public class task033 extends Customer {**

**void billing(){**

**String items = "onions";**

**int cost = 30;**

**super.items = "Potatoes";**

**super.cost = 50;**

**super.purchage\_list();**

**System.out.println(items);**

**System.out.println(cost);**

**System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**

**System.out.println(super.items);**

**System.out.println(super.cost);**

**// return 0;**

**}**

**public static void main(String[] args){**

**Customer cobj =new Customer();**

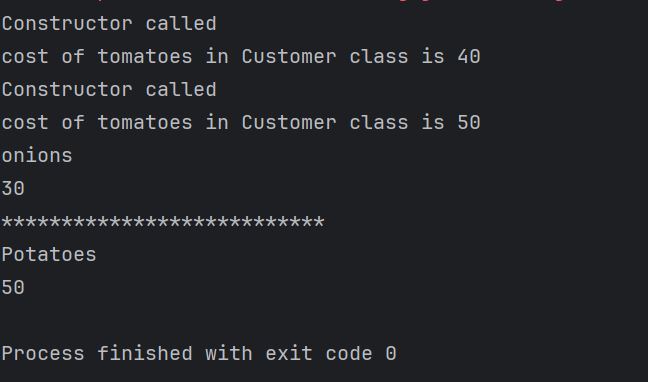
**cobj.purchage\_list();**

**task033 tobj = new task033();**

**tobj.billing();**

**}**

**}**

****

**Task 034 No of parameters**

**public class task034\_1 {**

**// Method with 2 parameters**

**void add(int x, int y) {**

**System.out.println("x = " + x + ", y = " + y);**

**}**

**// Method with 3 parameters (overloaded)**

**void add(int x, int y, int z) {**

**System.out.println("x = " + x + ", y = " + y + ", z = " + z);**

**}**

**public static void main(String[] args) {**

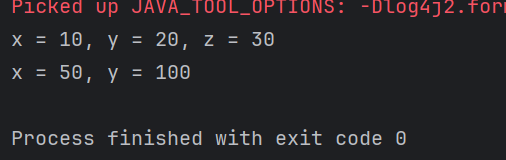
**task034\_1 obj = new task034\_1();**

**obj.add(10, 20, 30);**

**obj.add(50, 100);**

**}**

**}**

****

**Task 035 Type of parameters**

**public class task035 {**

**// Method with char parameters**

**void add(char x, char y) {**

**System.out.println("x = " + x + ", y = " + y);**

**}**

**// Method with int parameters (overloaded)**

**void add(int x, int y) {**

**System.out.println("x = " + x + ", y = " + y);**

**}**

**public static void main(String[] args) {**

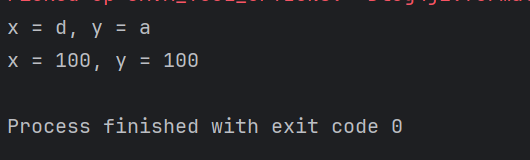
**task035 obj = new task035();**

**obj.add('d', 'a'); // calls char version**

**obj.add(100, 100); // calls int version**

**}**

**}**

****

**Task 036 Sequence of parameters**

**public class task036 {**

**void display(String name, int age) {**

**System.out.println("Name: " + name + ", Age: " + age);**

**}**

**void display(int age, String name) {**

**System.out.println("Age: " + age + ", Name: " + name);**

**}**

**public static void main(String[] args) {**

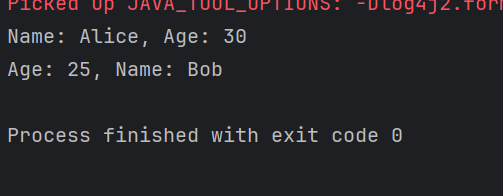
**task036 obj = new task036();**

**obj.display("Alice", 30); // matches String, int**

**obj.display(25, "Bob"); // matches int, String**

**}**

**}**

****

**Task 037 Encapsulation**

**class Employee {**

**private int pwd;**

**protected int salary;**

**public int empid;**

**// Constructor**

**Employee() {**

**System.out.println("Employee constructor called");**

**}**

**}**

**class task037 extends Employee {**

**task037() {**

**super(); // Call parent constructor**

**// pwd = 123;**

**salary = 50000; // ✅ protected: accessible in child class**

**empid = 10001; // ✅ public: accessible everywhere**

**}**

**public static void main(String[] args) {**

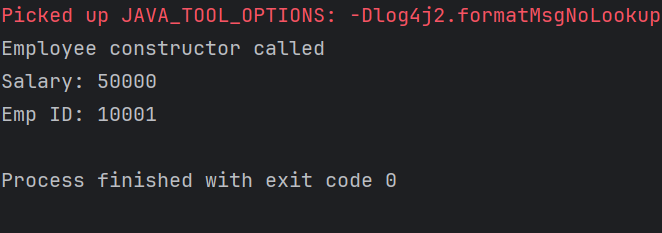
**task037 obj = new task037();**

**System.out.println("Salary: " + obj.salary);**

**System.out.println("Emp ID: " + obj.empid);**

**}**

**}**

****

**Task 38 ⇒ error code**

**public class AbstractDemo {**

**public static void main(String [] args) {**

**/\* Following is not allowed and would raise error \*/**

**Employee e = new Employee("George W.", "Houston, TX", 43);**

**System.out.println("\n Call mailCheck using Employee reference--");**

**e.mailCheck();**

**}**

**}**

**abstract class Employee {**

**private String name;**

**private String address;**

**private int number;**

**public Employee(String name, String address, int number) {**

**System.out.println("Constructing an Employee");**

**this.name = name;**

**this.address = address;**

**this.number = number;**

**}**

**public double computePay() {**

**System.out.println("Inside Employee computePay");**

**return 0.0;**

**}**

**public void mailCheck() {**

**System.out.println("Mailing a check to " + this.name + " " + this.address);**

**}**

**public String toString() {**

**return name + " " + address + " " + number;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String newAddress) {**

**address = newAddress;**

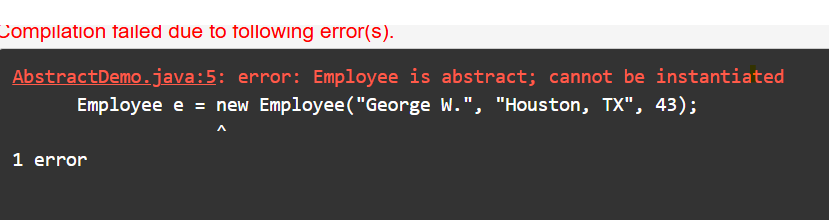
**}**

**public int getNumber() {**

**return number;**

**}**

**}**

****

**Task 39**

**// Abstract class**

**abstract class Employees {**

**private String name;**

**private String address;**

**private int number;**

**public Employees(String name, String address, int number) {**

**System.out.println("Constructing an Employee");**

**this.name = name;**

**this.address = address;**

**this.number = number;**

**}**

**public double computePay() {**

**System.out.println("Inside Employee computePay");**

**return 0.0;**

**}**

**public void mailCheck() {**

**System.out.println("Mailing a check to " + this.name + " " + this.address);**

**}**

**public String toString() {**

**return name + " " + address + " " + number;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getAddress() {**

**return address;**

**}**

**public void setAddress(String newAddress) {**

**address = newAddress;**

**}**

**public int getNumber() {**

**return number;**

**}**

**}**

**// Concrete subclass**

**class SalaryEmployee extends Employees {**

**public SalaryEmployee(String name, String address, int number) {**

**super(name, address, number);**

**}**

**@Override**

**public double computePay() {**

**return 1000.00; // Custom implementation**

**}**

**}**

**// Main class**

**public class AbstractDemo {**

**public static void main(String[] args) {**

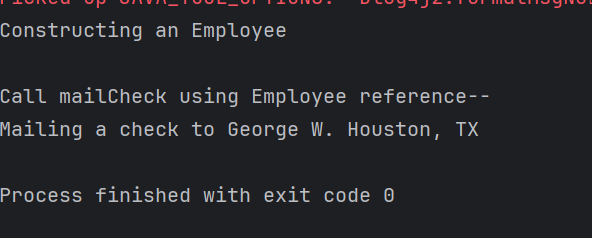
**Employees e = new SalaryEmployee("George W.", "Houston, TX", 43);**

**System.out.println("\nCall mailCheck using Employee reference--");**

**e.mailCheck();**

**}**

**}**

****

**Task 40**

**// Abstract class**

**abstract class Gadgets {**

**abstract void turnOn();**

**abstract void turnOff();**

**}**

**// Concrete implementations**

**class TVRemote extends Gadgets {**

**@Override**

**void turnOn() {**

**System.out.println("TV is turned ON.");**

**}**

**@Override**

**void turnOff() {**

**System.out.println("TV is turned OFF.");**

**}**

**}**

**class ACRemote extends Gadgets {**

**@Override**

**void turnOn() {**

**System.out.println("AC is turned ON.");**

**}**

**@Override**

**void turnOff() {**

**System.out.println("AC is turned OFF.");**

**}**

**}**

**class FanRemote extends Gadgets {**

**@Override**

**void turnOn() {**

**System.out.println("Fan is turned ON.");**

**}**

**@Override**

**void turnOff() {**

**System.out.println("Fan is turned OFF.");**

**}**

**}**

**class CoolerRemote extends Gadgets {**

**@Override**

**void turnOn() {**

**System.out.println("Cooler is turned ON.");**

**}**

**@Override**

**void turnOff() {**

**System.out.println("Cooler is turned OFF.");**

**}**

**}**

**// Main class**

**public class task040 {**

**public static void main(String[] args) {**

**Gadgets tv = new TVRemote();**

**Gadgets ac = new ACRemote();**

**Gadgets fan = new FanRemote();**

**Gadgets cooler = new CoolerRemote();**

**tv.turnOn();**

**tv.turnOff();**

**ac.turnOn();**

**ac.turnOff();**

**fan.turnOn();**

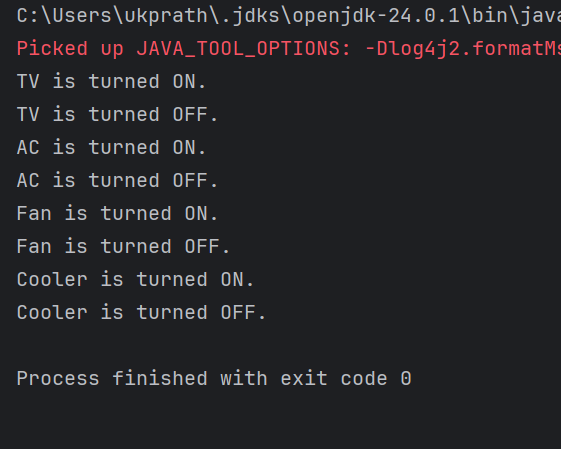
**fan.turnOff();**

**cooler.turnOn();**

**cooler.turnOff();**

**}**

**}**

****

**Task 41**

**interface testInterface {**

**final int tax = 10; // The constant is named 'tax'**

**void display();**

**}**

**class TestClass implements testInterface {**

**@Override**

**public void display(){**

**System.out.println("Myclass");**

**}**

**}**

**class Myclass**

**{**

**public static void main(String[] args)**

**{**

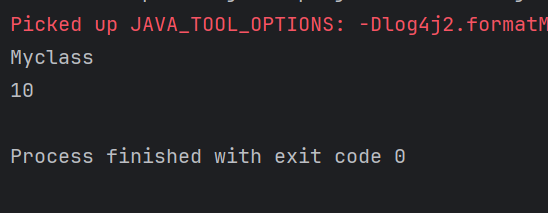
**TestClass t = new TestClass();**

**t.display();**

**System.out.println(t.tax); // ✅ This will work.**

**}**

**}**

****

**Driverclass Example**

**File 1**

**class Superclasss{**

**Superclasss(){**

**System.out.println("super class constructor called");**

**}**

**void superMethod(){**

**System.out.println("superMethod called");**

**}**

**}**

**File 2**

**public class Task112{**

**public static void main(String[] args){**

**System.out.println("Driver class called");**

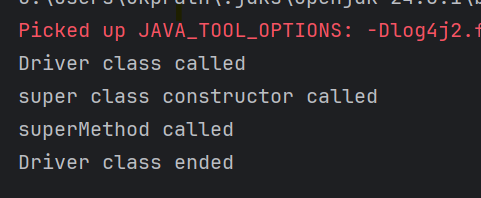
**Superclasss sobj = new Superclasss();**

**sobj.superMethod();**

**System.out.println("Driver class ended");**

**}**

**}**

****

**HOME WORK TASK**

**Task 21**

**public class task021{**

**public static void main(String[] args)**

**{**

**// initializing array**

**int[] arr = { 1, 2, 3, 4, 5 };**

**// size of array**

**int n = arr.length;**

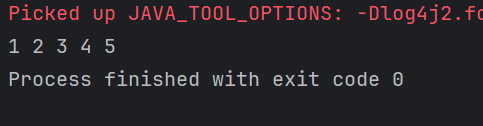
**// traversing array**

**for (int i = 0; i < n; i++)**

**System.out.print(arr[i] + " ");**

**}**

**}**

****

**TASK 022**

**public class task022 {**

**public static void main(String[] args)**

**{**

**// declares an Array of integers.**

**int[] arr;**

**// allocating memory for 5 integers.**

**arr = new int[5];**

**// initialize the elements of the array**

**// first to last(fifth) element**

**arr[0] = 10;**

**arr[1] = 20;**

**arr[2] = 30;**

**arr[3] = 40;**

**arr[4] = 50;**

**// accessing the elements of the specified array**

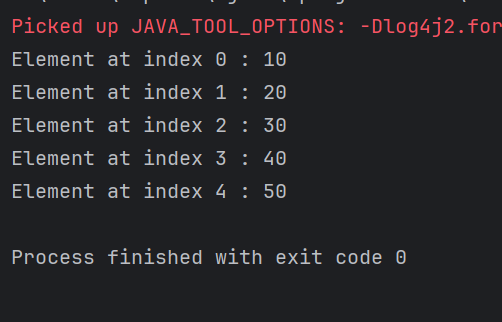
**for (int i = 0; i < arr.length; i++)**

**System.out.println("Element at index "**

**+ i + " : " + arr[i]);**

**}**

**}**

****

**Task023**

**class Student {**

**public int roll\_no;**

**public String name;**

**// Parameterized constructor**

**Student(int roll\_no, String name) {**

**this.roll\_no = roll\_no;**

**this.name = name;**

**}**

**// Optional: Define default constructor if needed**

**Student() {**

**this.roll\_no = 0;**

**this.name = "unknown";**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**// Optional: Create some Student objects**

**Student sobj1 = new Student();**

**Student sobj2 = new Student();**

**Student sobj3 = new Student();**

**// Declare and allocate memory for 5 Student objects**

**Student[] arr = new Student[5];**

**// Initialize each element with values**

**arr[0] = new Student(1, "aman");**

**arr[1] = new Student(2, "vaibhav");**

**arr[2] = new Student(3, "shikar");**

**arr[3] = new Student(4, "dharmesh");**

**arr[4] = new Student(5, "mohit");**

**// Access and print each element**

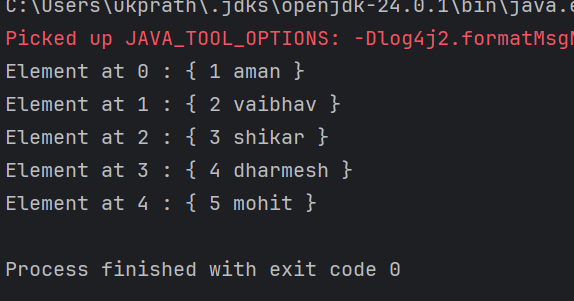
**for (int i = 0; i < arr.length; i++)**

**System.out.println("Element at " + i + " : { " +**

**arr[i].roll\_no + " " + arr[i].name + " }");**

**}**

**}**

****

**Task024**

**class Students{**

**public String name;**

**Students(String name){**

**this.name = name;**

**}**

**@Override**

**public String toString(){**

**return name;**

**}**

**}**

**public class task024{**

**public static void main (String[] args){**

**// declares an Array and initializing the**

**// elements of the array**

**Students[] myStudents = new Students[]{**

**new Students("Dharma"),new Students("sanvi"),**

**new Students("Rupa"),new Students("Ajay")**

**};**

**// accessing the elements of the specified array**

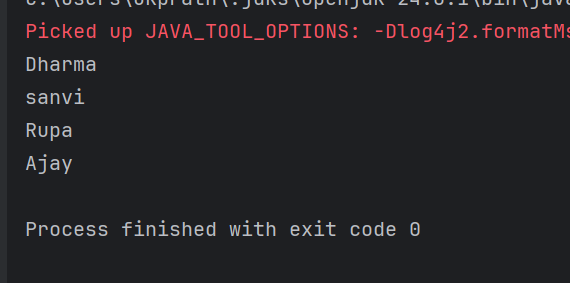
**for(Students m:myStudents){**

**System.out.println(m);**

**}**

**}**

**}**

****

**Task 25**

**public class task025 {**

**public static void main(String[] args)**

**{**

**int[] arr = new int[4];**

**arr[0] = 10;**

**arr[1] = 20;**

**arr[2] = 30;**

**arr[3] = 40;**

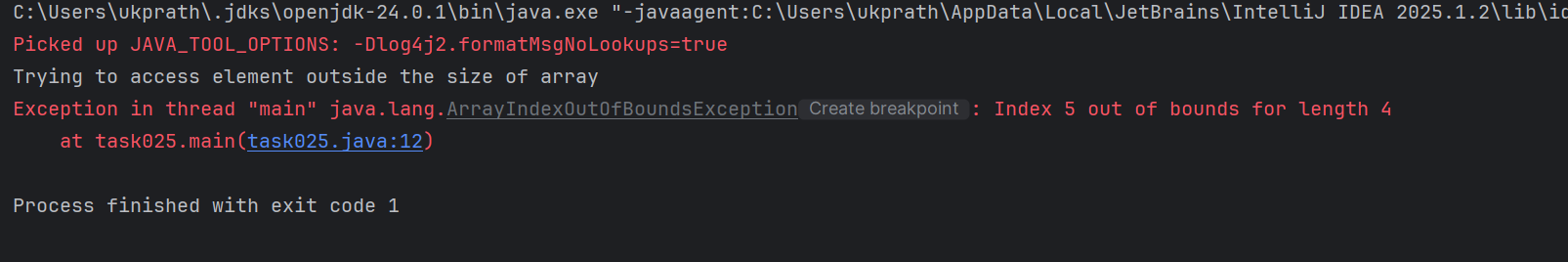
**System.out.println(**

**"Trying to access element outside the size of array");**

**System.out.println(arr[5]);**

**}**

**}**

****

**Task 26**

**class task026 {**

**public static void main(String[] args){**

**// Two Dimensional Array**

**// Declared and Initialized**

**int[][] arr = new int[3][3];**

**// Number of Rows**

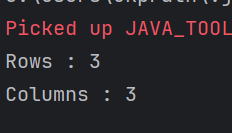
**System.out.println("Rows : " + arr.length);**

**// Number of Columns**

**System.out.println("Columns : " + arr[0].length);**

**}**

**}**

****

**Task 27**

**// Java Program to Multidimensional Array**

**// Driver Class**

**public class task027 {**

**// main function**

**public static void main(String args[])**

**{**

**// declaring and initializing 2D array**

**int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };**

**// printing 2D array**

**for (int i = 0; i < 3; i++) { // rows**

**for (int j = 0; j < 3; j++) // columns**

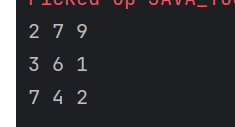
**System.out.print(arr[i][j] + " ");**

**System.out.println();**

**}**

**}**

**}**

****

**Task 28**

**public class task028 {**

**// Driver method**

**public static void main(String args[])**

**{**

**int arr[] = { 3, 1, 2, 5, 4 };**

**// passing array to method m1**

**sum(arr);**

**}**

**public static void sum(int[] arr)**

**{**

**// getting sum of array values**

**int sum = 0;**

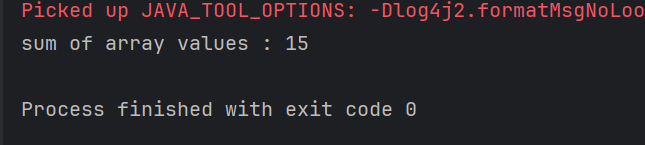
**for (int i = 0; i < arr.length; i++)**

**sum += arr[i];**

**System.out.println("sum of array values : " + sum);**

**}**

**}**

****

**Task 28\_1**

**// Java program to demonstrate**

**// return of array from method**

**class task028\_1 {**

**// Driver method**

**public static void main(String args[])**

**{**

**int arr[] = m1();**

**for (int i = 0; i < arr.length; i++)**

**System.out.print(arr[i] + " ");**

**}**

**public static int[] m1()**

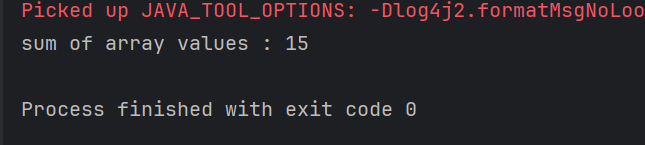
**{**

**// returning array**

**return new int[] { 1, 2, 3 };**

**}**

**}**

****

**Task 29**

**// Java program to demonstrate**

**// cloning of one-dimensional arrays**

**class clone {**

**public static void main(String args[])**

**{**

**int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };**

**int cloneArray[][] = intArray.clone();**

**// will print false**

**System.out.println(intArray == cloneArray);**

**// will print true as shallow copy is created**

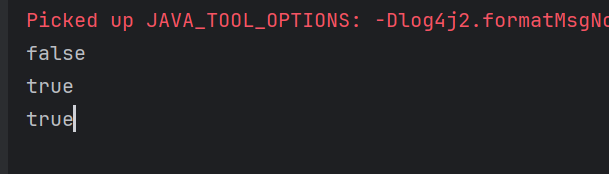
**// i.e. sub-arrays are shared**

**System.out.println(intArray[0] == cloneArray[0]);**

**System.out.println(intArray[1] == cloneArray[1]);**

**}**

**}**

****

**Task 30 shallow copy , Deep copy**

**import java.util.Arrays;**

**public class task30 {**

**public static void main(String[] args) {**

**int[] ar1 = {1, 2, 3};**

**int[] ar2 = {4, 5, 6};**

**int[][] arr = {ar1, ar2};**

**// Shallow copy (outer array copied, inner arrays shared)**

**int[][] shallowCopy = Arrays.copyOf(arr, arr.length);**

**// Deep copy (both outer and inner arrays copied)**

**int[][] deepCopy = new int[arr.length][];**

**for (int i = 0; i < arr.length; i++) {**

**deepCopy[i] = Arrays.copyOf(arr[i], arr[i].length);**

**}**

**// Modify original inner array**

**ar1[0] = 99;**

**// 🖨️ Results**

**System.out.println("Original array: " + Arrays.toString(arr[0]));**

**System.out.println("Shallow copy: " + Arrays.toString(shallowCopy[0]));**

**System.out.println("Deep copy: " + Arrays.toString(deepCopy[0]));**

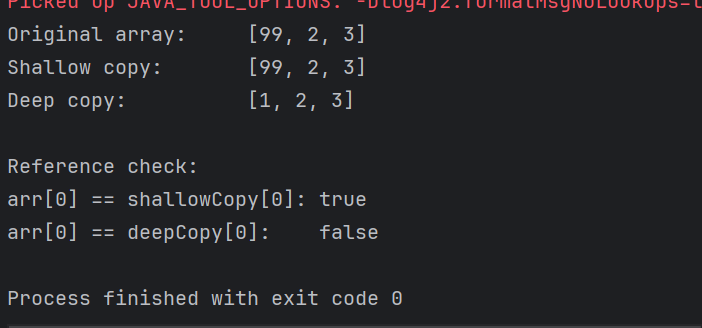
**System.out.println("\nReference check:");**

**System.out.println("arr[0] == shallowCopy[0]: " + (arr[0] == shallowCopy[0])); // true**

**System.out.println("arr[0] == deepCopy[0]: " + (arr[0] == deepCopy[0])); // false**

**}**

**}**

****